XXV Jornadas de Automática
Ciudad Real, del 8 al 10 de septiembre de 2004

APLICACIÓN CLIENTE/SERVIDOR MULTIPLATAFORMA EN INTERNET PARA LABORATORIO VIRTUAL DE ROBÓTICA

E. Martínez Cámara, E. Jiménez Macías
Departamento de Ingeniería Eléctrica. Área de Ingeniería de Sistemas y Automática.
Universidad de La Rioja. C/Luis de Ulloa 20 CP-26004 Logroño (La Rioja)
e-mail: emilio.jimenezm@die.unirioja.es; eduardo.martin@alum.unirioja.es

F. Sanz Adán, J. Santamaría Peña
Departamento de Ingeniería Mecánica. Área de Expresión Gráfica en la Ingeniería.
Universidad de La Rioja. C/Luis de Ulloa 20 CP-26004 Logroño (La Rioja)
e-mail: {felix.sanz;jacinto.santamaría}@dim.unirioja.es

M. Pérez de la Parte
Departamento de Ingeniería de Sistemas y Automática. Área de Ingeniería de Sistemas y Automática.
Universidad de Sevilla. Avda. de los Descubrimientos SN, CP-41092 Sevilla
e-mail: mpparte@cartuja.us.es

Resumen

La dificultad de acceder a sistemas robóticos reales a un precio asequible lleva a la necesidad de desarrollar aplicaciones a medida para facilitar la interactuación con sistemas automatizados que incluyan robots. Con esta idea se plantea el desarrollo de una aplicación cliente/servidor para la simulación gráfica de robots por ordenador en entornos virtuales. Además el programa debe mostrar un correcto funcionamiento en entorno multiplataforma, así como una interface compatible con Internet, para permitir un acceso generalizado a los recursos disponibles, con vistas a un laboratorio virtual. Esta ejecución a través de Internet está sujeta al soporte externo que aporta casi cualquier navegador Web existente hoy en día en el mercado. La aplicación que se presenta, Simrobot3D, ha sido desarrollada con esas características en la Universidad de La Rioja.

Palabras Clave: Aplicación cliente/servidor, simulación 3D, entorno multiplataforma, laboratorio virtual, robots.

1 INTRODUCCIÓN

En este artículo se presenta el desarrollo de una aplicación cliente/servidor a medida, para la simulación gráfica de robots por ordenador en escenarios virtuales, con funcionamiento en entorno multiplataforma, así como una interface compatible con Internet, para permitir un acceso generalizado a los recursos disponibles (Figura 1).

Esta capacidad de difusión a través de la red se consideraba de gran importancia, ya que se ajusta perfectamente a una tendencia cada vez con mayor auge en la comunidad universitaria actual, como es la enseñanza virtual. De hecho un número cada vez más importante de universidades que, tanto en España como en el resto del mundo, mantiene en funcionamiento distintos cursos e incluso carreras de forma virtual, crece incesantemente. Si bien es cierto que estas actividades se centran fundamentalmente en el campo de las letras, como en el caso de la Universidad de La Rioja, no se debe descartar la posibilidad de que poco a poco vaya introduciéndose también en el campo de las ciencias y más concretamente en las carreras científico-técnicas, al menos como una ayuda o un apoyo importante a la enseñanza presencial.

Por otro lado, industrialmente se puede apreciar también una disposición a la conexión de las fábricas, las plantas y los distintos elementos de las cadenas de montaje a la red. Desde este punto de vista, el programa de simulación desarrollado podría emplearse como parte de un paquete de software para el control de robots a través de la red.

Desde un punto de vista más funcional la aplicación es capaz leer un fichero de datos con todas las especificaciones de geometría y funcionamiento que definen, tanto el robot o robots a simular, como cualquier otro elemento que también entre dentro del escenario de simulación.

Con esta base, y una vez la aplicación tiene los datos concretos de la simulación, éstos se presentan gráficamente en pantalla dentro del entorno tridimensional aportado por Java3D. Además el

usuario tiene total libertad para interactuar con los modelos tridimensionales de los robots creados.

Otro objetivo básico que se pretende alcanzar es el de dotar a la aplicación de la capacidad de comunicación cliente-servidor, con el fin de realizar un sistema de telerrobótica. Es decir, se busca que el usuario sea capaz de simular un escenario virtual y además, si así lo desea, ver el resultado de esta simulación en un servidor remoto. Mediante la habilitación de este servidor remoto se aporta capacidad para simular y controlar motores, así como, la posibilidad de modificar de forma dinámica los parámetros de los mismos en tiempo de ejecución.

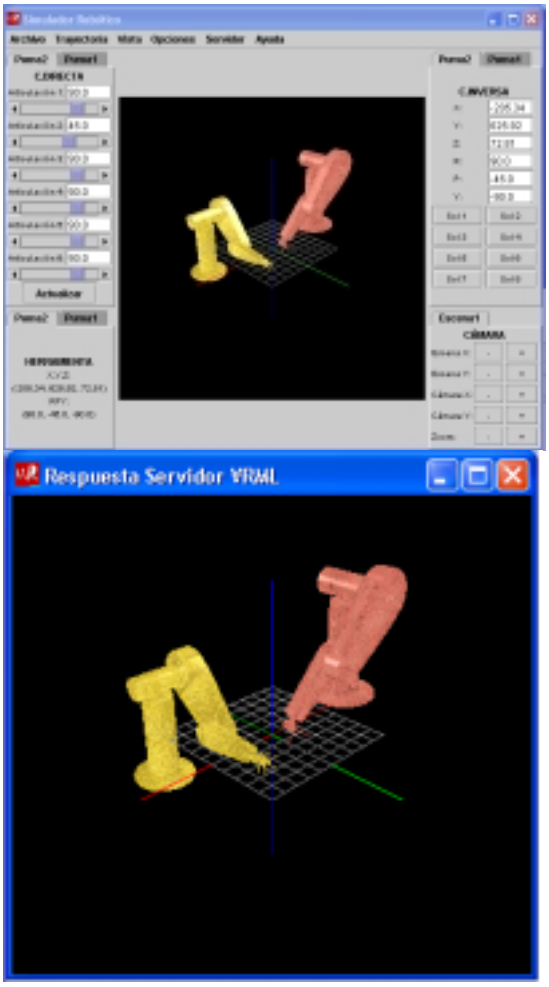

Figura 1: Aplicación SimRobot3D (arriba) en conexión con el servidor remoto (abajo).

2 ALTERNATIVAS Y SOLUCIONES ADOPTADAS

En primer lugar es necesario crear un compilador, de tal forma que a partir de un fichero correctamente estructurado, y con un lenguaje concreto previamente definido, el programa sea capaz de entender lo que el usuario ordena para actuar en consecuencia [6].

Desarrollar un compilador no es tarea fácil pero existen varias herramientas que facilitan en gran medida la labor. Por un lado están los analizadores léxicos y por otro los analizadores sintácticos, también conocidos como compiladores de compiladores. Estas herramientas permiten crear el código necesario para analizar léxicamente un texto, obteniendo las "palabras" que lo forman (más concretamente las estructuras de caracteres que se definan previamente y que reciben normalmente el nombre de *tokens*). Y en función de estas "palabras" pueden realizar el análisis sintáctico (cómo están estructuradas y ordenadas) y actuar de la forma oportuna.

Existe una gran variedad de analizadores léxicos para cada uno de los lenguajes de programación posibles. Todos manejan una estructura y características similares al LEX creado para código C. En este caso se ha optado por el JFLEX versión 1.3. Este programa presenta como principal característica una mayor velocidad en los analizadores léxicos que crea. También existe una gran variedad de analizadores sintácticos. Al igual que en el caso anterior la mayoría deriva del YACC (Yet Another Compiler Compiler) manteniendo las características básicas. En este caso se ha seleccionado el CUP versión 0.10j, sobre todo por su compatibilidad con el analizador léxico JFLEX.

En cuanto a la representación gráfica de escenarios tridimensionales en la Web [5] existen básicamente tres tecnologías: Java3D, VRML y X3D. X3D es actualmente una tecnología en desarrollo bajo los estándares de XML y representa la evolución de VRML [1] llevada a cabo por el VRML Consortium (Web3d Consortium en la actualidad). En vista de esto se ha seleccionado como tecnología de base para la representación gráfica Java3D, pero también se ha dado soporte a VRML bajo la cobertura de Java3D. De esta forma es posible exportar el escenario a formato VRML para su posterior tratamiento en cualquier editor gráfico que soporte VRML.

Otros aspectos que es necesario tener en cuenta son todos los que se derivan de la implementación de la estructura cliente-servidor. Básicamente son dos puntos diferenciados: dotar al applet de capacidad de conexión con servidores y desarrollar e implementar

un server que se ajuste a las necesidades concretas de la aplicación.

En lo que respecta a la capacidad de los applet para realizar conexiones con servidores remotos, es necesario conocer las limitaciones impuesta por la máquina virtual java (JVM – Java Virtual Machine) en la ejecución de código de confianza (trusted) o no (untrusted). Más concretamente la ejecución de código untrusted no permite la conexión con servidores distintos al propio servidor de origen del código; tampoco permite la lectura y escritura de fichero locales, etc. Para conseguir que la JVM considere el código como trusted es necesario que el usuario final acepte este código como de confianza. Una de las formas de conseguirlo es firmar el applet previamente empaquetado en un fichero JAR.

Por último, para completar la estructura cliente-servidor es necesario dotar al sistema de un server con capacidad para comunicarse con el cliente y ejecutar las funciones de simulación y control de los motores presentes en el escenario virtual generado por la aplicación cliente. Dentro las posibles soluciones para solventar este problema existen múltiples posibilidades, pero de forma general se pueden dividir en dos categorías: creación de un sistema propio o uso de aplicaciones existentes con capacidad de ejecución en modo server. La creación de un server propio no plantea problemas ya que existen funciones y librerías destinadas a la creación de los mismos en Java, C++, etc. El principal problema es la limitación y restricción que plantean en cuanto a capacidades de simulación y control, así como en las posibilidades por parte del cliente de realizar sus propias modificaciones en los sistemas de control.

Debido a estas carencias se estudio también la posibilidad de usar algún software comercial con capacidad de simulación de sistemas dinámicos y que se adaptase a las necesidades concretas de la aplicación. Existen varios softwares que se podrían ajustar con mayor o menor esfuerzo a los requerimientos del sistema: Sysquake de Calerga, LabView de National Instruments, Matlab de MathWorks, etc. Pero, sin lugar a dudas, el más sencillo y aquél del que se disponen mayores conocimientos prácticos es Matlab. Además, como es sabido, Matlab dispone de librerías centradas en los sistemas de control, así como de una aplicación específica para la simulación de sistemas Simulink.

Con esta base y los conocimientos previos adquiridos sólo queda estudar si se puede conectar a través de Internet con Matlab-Simulink. Por defecto Matlab no permite la conexión en red, pero hay varias opciones para solventar este problema. Por un lado se puede usar una librería externa desarrollado por el profesor Werner Zimmermann de la Universidad de Ciencias Aplicadas de Esslingen (Alemania) llamada IOLib (Hardware Input / Output Library for Matlab / Simulink). Los principales problemas de esta librería son la limitación en cuanto el número de entradas y salidas que puede soportar y la imposibilidad de crear sistemas de simulación al instante en función de los requerimientos del cliente. Por otro lado se puede utilizar una toolbox propia de Matlab llamada Matlab Web Server. Esta toolbox está específicamente diseñada para permitir la creación de páginas web con capacidad para comunicarse con Matlab y ejecutar unos ficheros.m concretos específicamente diseñados para ese fin.

Aunque de las dos soluciones estudiadas la más fácil de implementar es la utilización de la librería IOLib, finalmente se ha utilizado la toolbox Matlab Web Server ya que permite un mayor control del entorno de Matlab.

3 CREACIÓN DE ESCENARIOS

La definición del escenario virtual del simulador se realiza a través de un fichero de texto creado por el usuario (ver Figura 2). Este fichero debe contener toda la información necesaria para la creación del escenario y el correcto funcionamiento de los distintos elementos que lo componen: desde todo lo que respecta al apartado gráfico hasta la [4] cinemática de los robots presentes en la escena. En el apartado gráfico es esencial llevar a cabo correctamente la definición de los vértices que delimitan las formas geométricas a presentar en el simulador. También se deben detallar cuáles son estas formas geométricas y a qué objeto, más concretamente a qué eslabón de qué objeto, están ligadas. En lo que hace referencia a la cinemática de los objetos presentes en la escena, se emplea la relación existente entre los distintos eslabones del objeto definida mediante los parámetros de Denavit - Hartenberg. Además para el correcto funcionamiento de la cinemática inversa es ineludible especificar una concreta y previamente definida en el conjunto de la aplicación. Es decir, que la cinemática inversa a aplicar tiene que existir como una clase más del simulador y estar correctamente definida.

Como se puede ver, la creación de una escenario es una tarea compleja que requiere conocer los elementos gráficos que maneja el simulador, la estructura del escenario, el lenguaje a utilizar y las características de los robots que se quiera simular. Los elementos gráficos disponibles en la aplicación son tres: polígono, caja y cilindro.

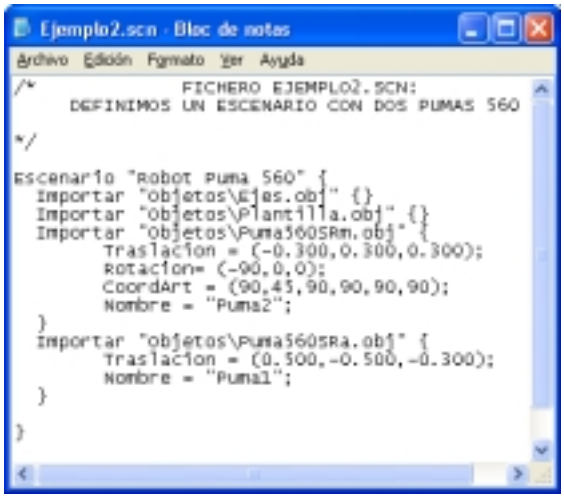

Figura 2: Ejemplo de fichero de definición de escenario

3.1 ESTRUCTURA DEL ESCENARIO

El fichero de texto que defina el escenario debe estar formado por un escenario con su nombre correspondiente que englobe a todo lo demás mediante unas llaves. Dentro de este escenario se encuentran los distintos objetos del mismo. A su vez cada objeto queda determinado por los distintos eslabones que lo forman. Y por último, cada eslabón se concreta con sus parámetros de Denavit-Hartenberg y las formas geométricas que le correspondan.

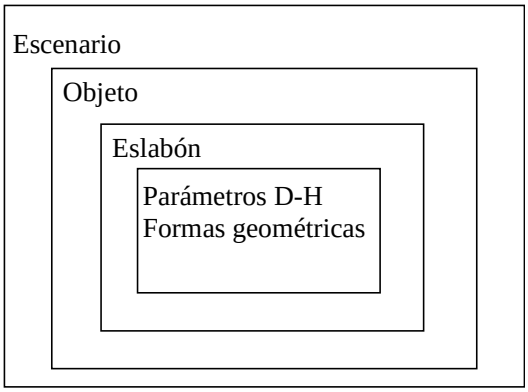

Figura 3: Estructura Básica del escenario

La Figura 3 intenta reflejar la idea expuesta anteriormente y dejar patente el proceso de composición seguido para conseguir finalmente definir todo el escenario en función de unas simples forma geométricas.

Para facilitar la tarea de creación del escenario se ha introducido la posibilidad de importar objetos previamente definidos en otros ficheros de texto, sin necesidad de volver a copiar todas las especificaciones del objeto en el fichero del escenario. Por ejemplo, si se ha diseñado un objeto simple con forma de caja de unas dimensiones concretas y se quiere hacer aparecer dos veces en el mismo escenario, no es necesario escribir dos veces la definición de la caja en el escenario, basta con importarla dos veces. A la hora de importar objetos se les puede aplicar una serie de modificaciones: traslación, rotación, cambio de coordenadas articulares y cambio de nombre. De esta forma dos objetos idénticos en cuanto a definición, dos pumas 560 por ejemplo, se pueden colocar en la misma escena pero ocupando posiciones diferentes, con orientaciones distintas, otras coordenadas articulares y otros nombres.

Para poder importar objetos es imprescindible que éstos estén correctamente definidos en un fichero de texto independiente (Figura 4). De este modo se distinguen dos tipos de ficheros: los destinados a la creación de escenarios, a los cuales se les a denotado con una extensión .SCN, y los relacionados con la definición de objetos pertenecientes a los escenarios, con extensión .OBJ.

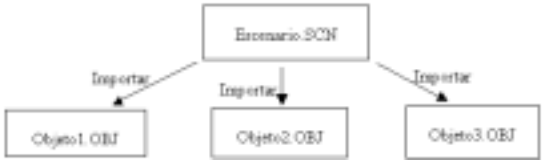

Figura 4: Modelo de importación de objetos para la creación de escenarios

4 MODELO DEL ROBOT

La creación del escenario en [3]3D, no es otra cosa que la traslación del modelo definido por el usuario (en base a las reglas establecidas anteriormente) a un modelo entendible por Java3D, conforme a sus reglas y estructuras. Java3D presenta una estructura arbórea para definir el universo virtual. Dentro de esta estructura es la rama de contenidos la que se debe adecuar a la definición de escenario realizada por el usuario, de tal forma que se sea capaz de reflejar de forma directa todas las definiciones de objetos, robots, rotaciones, traslaciones, etc. que el usuario pueda introducir en el escenario.

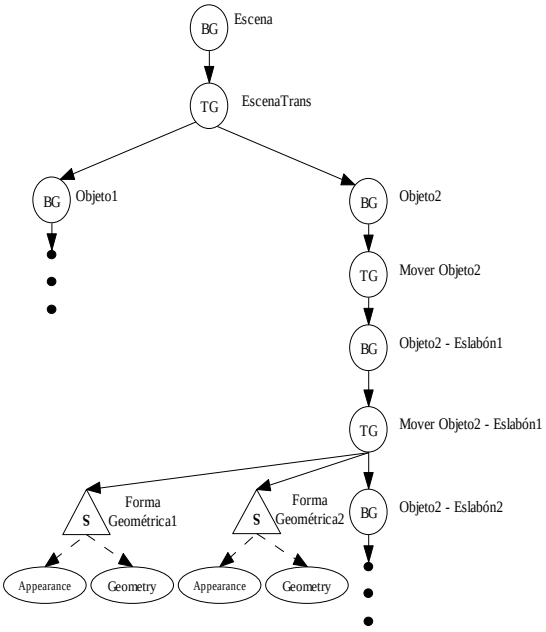

Figura 5: Diagrama genérico de escenario en Java3D

Para alcanzar este objetivo se ha establecido una estructura genérica del escenario en Java3D, (ver Figura 5). Como se puede apreciar todo el escenario queda englobado por un nodo *Escena* que será el elemento fundamental de la rama de contenidos del árbol del universo virtual de Java3D [2]. Además de este nodo *Escena* también es necesario añadir a la rama de contenidos, una serie de nodos de iluminación ambiental y direccional para dar realismo, y perspectiva al universo virtual. Pero donde realmente se encuentra el contenido es en el nodo Escena y en sus hijos.

Un vistazo rápido a la estructura de dicho nodo permite observar una composición de un BranchGroup, un TransformGroup, un BranchGroup, un TransformGroup... desde el primer nivel. Esta estructura posibilita la creación y el control de todos los elementos de la escena. De forma genérica se puede decir que de un BranchGroup, que define el Eslabón, Objeto o Escenario, cuelga directamente un TransformGroup que va a permitir el control de todos los nodos que configuran, definen y dependen de ese elemento.

5 SISTEMA SERVIDOR

5.1 MATLAB WEB SERVER

Matlab Web Server permite al desarrollador crear aplicaciones que aprovechan la capacidad de comunicación de Internet. De esta forma se pueden enviar datos de entrada a Matlab para que realice los cálculos u operaciones definidas y que, posteriormente, devuelva los resultados y se muestren en pantalla mediante un navegador Web.

En su configuración más simple el sistema (ver Figura 6) puede constar de: un navegador Web ejecutándose en un ordenador Cliente y un servidor realizando las funciones de Matlab Web Server (matlabserver) y de servidor Web (httpd).

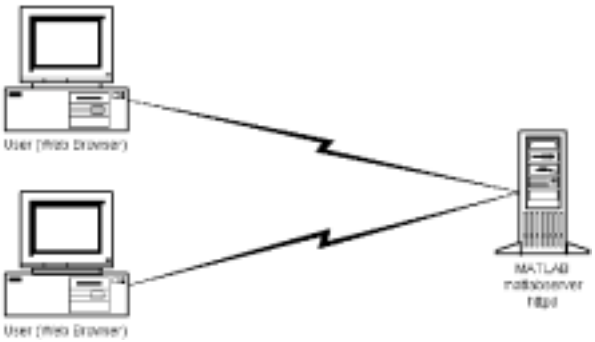

Figura 6: Configuración simple del sistema Matlab Web Server

Matlab Web Server presenta un potencial importante en el ámbito de la simulación on-line, pero también presenta problemas y limitaciones que reducen ese potencial.

Estas limitaciones hacen imposible el uso de Matlab Web Server tal y como está definido dentro del proyecto que se está desarrollando, ya que, básicamente, no es posible realizar una simulación y control de motores dinámica y en función de las características presentes en la escena. Esto es así, porque Matlab Web Server limita su ejecución a ficheros .m preestablecidos y no permite la ejecución directa de ficheros .m generados dinámicamente desde la aplicación cliente.

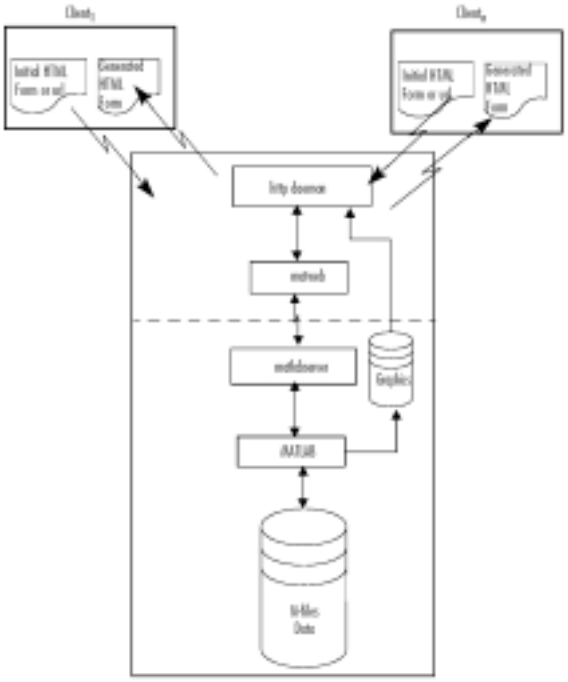

Figura 7: Esquema de funcionamiento de Matlab Web Server

Como se puede ver en el esquema (Figura 7) matlabserver mantiene una comunicación directa con MATLAB. Realmente la ejecución del fichero .m se realiza mediante matlabserver. Es decir matlabserver es el encargado de cargar una nueva ejecución de MATLAB y enviarle los comandos necesarios para procesar el fichero .m definido. Además matlabserver es un servidor TCP/IP que espera la conexión de cliente a través de un puerto definido por el administrador del sistema.

En este punto ya se sabe que matlabserver es el encargo de comunicar a MATLAB qué comandos debe ejecutar. Ahora sólo queda conocer quién establece cuáles son esos comandos, y cómo. Obviamente es el CGI matweb el encargo de analizar/filtrar los datos de entrada provenientes del formulario web y definir qué comandos se deben ejecutar. Una vez que matweb lo sabe se lo comunica a matlabserver que es quien realmente realiza el proceso de ejecución.

En definitiva, como se ha podido observar, matlabserver es el núcleo central del sistema de comunicación on-line de Matlab Web Server. Mientras que matweb es una capa a nivel del servidor Web para analizar e interpretar los datos de entrada. Incluso es posible comunicar el CGI matweb desde un servidor distinto al servidor en el que se encuentra y se ejecuta matlabserver y MATLAB.

De todo lo anterior se desprende que si es posible realizar las funciones del CGI y conseguir una comunicación directa con matlabserver desde la aplicación cliente, se podrá conseguir un aumento notable del potencial de Matlab Web Server para las simulaciones on-line.

Para alcanzar ese objetivo sólo se requiere conocer cuál es el protocolo de comunicación entre matweb y matlabserver. Conseguir este protocolo es cuestión de realizar un concienzudo análisis de las cadenas de comunicación de entrada y salida entre ambos programas. Finalmente el resultado obtenido es un protocolo simple basado en una cabecera, los datos y un fin de mensaje.

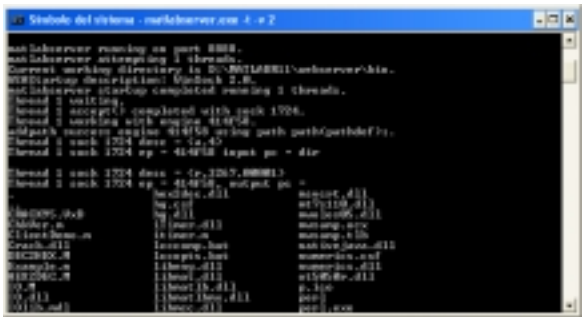

Figura 8: Matlabserver en ejecución

Implementando este protocolo en la aplicación cliente es posible realizar una comunicación directa con matlabserver y, por tanto, con MATLAB. De esta forma se dispone de total libertad para realizar cualquier tipo de operación dentro de la consola de comandos de MATLAB desde la aplicación cliente.

5.2 MATLAB-SIMULINK

Una vez que se tiene posibilidad de conectar directamente con Matlab desde la aplicación cliente sólo es necesario recoger los datos recibidos e implementar una simulación acorde a ellos. En la Figura 9 se puede ver el resultado final del modelo de control implementado para un escenario con 6 articulaciones (motores).

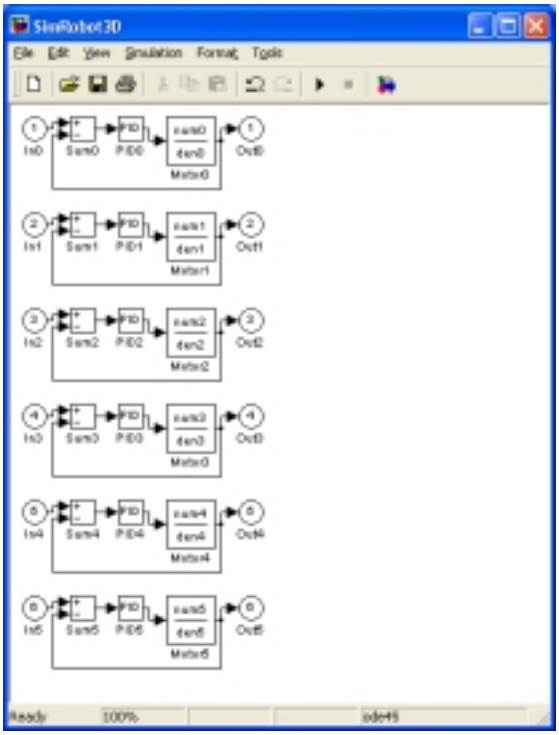

Figura 9: Ejemplo de un sistema de control para un escenario con 6 articulaciones

Se puede apreciar que el sistema de control es un sistema en lazo cerrado con un controlador PID, pero lo más relevante es el diseño del modelo en sí mismo. En definitiva, que el modelo generado no es modelo estático, sujeto a unas características concretas previamente definidas, sino que el modelo se genera dinámicamente en función de los parámetros introducidos por el usuario. Incluso el usuario puede ir modificando los parámetros en tiempo de ejecución.

Para conseguir esta libertad en cuanto a la definición del sistema en Matlab-Simulink, no se pueden emplear los típicos modelos (.mdl), ya que estos son ficheros estáticos que una vez creados y guardados sólo se pueden modificar a nivel local. La solución es generar dinámicamente el modelo de Matlab-Simulink en tiempo de ejecución y según las necesidades concretas del usuario. Para poder hacer esto desde una aplicación cliente sin acceso local al servidor de Matlab y sin acceso al interface gráfico de usuario de Matlab-Simulink, es necesario recurrir a la programación directa del modelo desde la línea de comandos de Matlab. Aunque Simulink es una toolbox más de Matlab, y como tal soporta programación directa desde la línea de comandos, no es lo normal, ni lo más sencillo llevar a cabo esta programación, ya que se pierde la referencia visual que aporta el interface gráfico de usuario.

Referencias

[1] Ames, A., Nadeau, D. & Moreland, J. (1997) VRML 2.0 sourcebook (2nd ed.). New York. John Wiley & Sons.

[2] David M. Geary, Graphic Java 1.2: Mastering the JFC. Volumen II: SWING. Sun Microsystems Press.

[3] D. Selman. (2002), Java 3D Programming. Independent Publishers Group.

[4] John J. Graig, Introduction to robotics: Mechanics and Control. Second Edition. Addison-Wesley Publishing Company.

[5] Walsh, A.E. and M. Bourges-Sevenier, (2001), Core Web3D. Prentice Hall PTR.

[6] W. M. Waite, Beyond LEX and YACC: How to Generate the Whole Compiler. Department of Electrical and Computer Engineering. University of Colorado. Boulder, Colorado.